NISTIR 6401

COMPUTER-INTEGRATED KNOWLEDGE SYSTEM (CIKS) NETWORK: REPORT OF THE 2ND WORKSHOP

Lawrence J. Kaetzel, Editor
K-Systems
Brownsville, MD

Building and Fire Research Laboratory
Gaithersburg, Maryland 20899

United States Department of Commerce
Technology Administration
National Institute of Standards and Technology

NISTIR 6401

COMPUTER-INTEGRATED KNOWLEDGE SYSTEM (CIKS) NETWORK: REPORT OF THE 2ND WORKSHOP

Lawrence J. Kaetzel, Editor
K-Systems
Brownsville, MD

December 1999

Building and Fire Research Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

An Intermediate Model for Interoperation of Web-distributed Applications With A Design Model

James Andrew Arnold
Stanford University

Abstract: *This paper defines the data and inference requirements for the interoperation of analysis applications and a product model. Application inputs often require sets of complex data that constitute database views of a product model. We introduce a method that works with the STEP and PLIB product description standards to define an intermediate model that assists interpretation (selects and extracts information), and validates views from a product model for the inputs of an application. This research built and tested the intermediate model framework in a software prototype, the Internet Broker for Engineering Services (IBES). It integrates applications that specify and diagnose faults in certain components, for example pumps and valves. Thus the research scope explores a sub-set of the general problem of integrating product data semantics between engineering applications. This research suggests that it is possible to define a general set of services to assist interpretation and validate information from a product model for an engineering purpose.*

Introduction

The World Wide Web (Web) and related network technologies comprise an infrastructure for ubiquitous information exchange and eventually, electronic commerce through access to markets of product information and software applications.

To date, Web applications add value primarily for communication and publishing. For example Email is prevalent and many applications exist to create Web pages. Applications for business transactions are in the early adoption phase. Examples include procurement, inventory control, and banking services. In the future Web applications will be available to automate core business functions and leverage them across organizations. This third class of software, often categorized as enterprise computing technology, will generate the greatest impact on the business process because it formalizes expert knowledge for integration and reuse between organizations. The presence of Electronic Data Interchange (EDI) and electronic banking services indicate that the representation requirements for the automation of basic business transactions are relatively well understood. However automation of core design and engineering tasks necessitates additional capability for modeling design content and reasoning to enable software interoperation.

For example, consider component selection for a design by using the Web to integrate a selection program supplied by an information publisher (a product vendor or dealer) directly with a product model in a CAD system developed or maintained by an information subscriber (process engineer or facility owner). Throughout this paper we will refer to a specific case of this problem involving the selection of a control valve for a process plant sub-system. Our example demonstrates automation of a routine engineering task that leverages knowledge between a publisher and subscriber. Furthermore, it highlights the functionality for modeling design content and reasoning that we believe is necessary to support task automation within an interoperable computing environment.

We use the term component to denote a complex (assembled part) item as opposed to a part (single item) that is normally manufactured and sold by a vendor and is incorporated into the facility. The term "component model" differentiates this information model from a "product model", a computer-readable description of design information that describes the facility itself. The test case draws from a textbook example for selecting the correct valve size under conditions of turbulent liquid flow.

> An eight-inch schedule 30 pipe has a flow of a maximum 1,600 gpm of river water. The pressure immediately upstream of the control valve is 27.9 psig, or 42.6 psia. The pressure in the pipe immediately downstream of the valve is 20 psig, or 34.7 psia. Select and size an appropriate control valve.[7]

The problem statement gives the design conditions of an operating case to compute the required coefficient of flow[18], *Cv* (required) = 569. If a design engineer wishes to use a 60-degree butterfly valve, a 6" nominal diameter valve satisfies the requirements. This solution is obtained by entering the design condition variables into publisher's product selection program, or by computing the *Cv* manually and inspecting the proper selection table in a publisher's product catalog.

Control valve selection is a typical engineering task that often requires information exchange between a product vendor and design engineer. While the solution for the example is simple, the problem statement abstracts the knowledge work, the context of information gathering, organization, analysis and evaluation that the task requires. Today, much of this information is located in non-integrated spreadsheets, equipment databases, CAD drawings, and project specifications.[2] In the future such information will be formalized in a product model.

This research tests the hypothesis that a general set of services can be defined to support interoperation and integration of a CAD system incorporating an underlying product model with an engineering analysis program. These services must select, extract, validate and communicate information from a product model for a software program. Figure 1 illustrates the research objective in terms of an IDEF0 diagram. The product model is the input to the integration activity. The output is the validated information for an application's input arguments. The constraint on the activity is the vocabulary that delineates the expressiveness of the intermediate model. The method for the integration activity consists of logic to support information selection, extraction, validation and communication. Generalization of the integration activity constraints and methods enables integration of distinct product models and engineering applications which, in principle, lowers the cost of integration between any given publisher and subscriber. We envisage these services as middle-ware that will be implemented on top of a distributed object protocol, such as the Common Object Request Broker (CORBA)[16] or the Microsoft ActiveX/Distributed Common Object Model (DCOM)[17], and function as an engineering content broker. Thus we named the proof-of-concept prototype the Internet Broker for Engineering Services (IBES).

This paper is organized as follows. The next section briefly summarizes related AEC product modeling research; Section 3 introduces the IBES conceptual framework and Section 4 furnishes a step by step description of the integration services provided by IBES. Each step is presented in terms of a process description for the test case and the underlying system functionality. The paper concludes with a brief description of the implementation platform used for the research and a summary discussion.

Related Work

Historically Architecture, Engineering, Construction (AEC) product-modeling research has focused on the semantic definition and structure of information models to describe facilities and exchange them between design consultants.[1,3,6] Other research has emphasized data structures and database management systems that support model evolution and change over the project life cycle.[8,9,10] Recent research emphasizes product model formulation as an outgrowth of a cognitive design process. Clayton describes a software prototype, called the Semantic Modeling Extension (SME), that provides a human computer interface and modeling environment for ‘interactive interpretation’, the user-initiated mapping of graphical features (forms) from a CAD drawing to a product model for design analysis relevant for architectural practice.[5] Our research explores computer automation of interpretation in the context of an existing product model and typical engineering analysis tasks.

Distinct representation requirements for the design disciplines are leading new modeling efforts to explore model interoperation. For example, the Industry Alliance for Interoperability has committed to publication of the Industry Foundation Classes (IFC) in the CORBA Interface Definition Language (IDL).[16] The IFCs define object hierarchies of building components and could eventually become the basis for a distributed AEC object environment. The Standards for the Exchange of Product data (STEP) community has begun to address STEP instance data access through languages and tools like EXPRESS-X[23] and the Standard Data Access Interface (SDAI) implementation in Java.[14] These tools enable the definition of static views and queries of a STEP model from distributed locations.

A reference library for components has become another important research topic for object interoperation. The ISO Parts Library Standard (PLIB) represents the leading effort to date in this area.[20,21,22] We selected the PLIB vocabulary to describe the semantics of the IBES intermediate model because of its resources for modeling complex engineering information independently of a design context. This paper describes just a few key elements of the PLIB conceptual model that are relevant to the IBES research.

PLIB defines a semantic dictionary that is populated by *dictionary elements*.[21] A dictionary element describes a semantic concept at any level of granularity, for example a component, a property of a component, or a view of a component from a product model. Entries in the dictionary are associated with an absolute identifier, called a *basic semantic unit* (BSU) that is sub-typed by identifier category, for example by publisher or component class. The dictionary supports relations between entries that permit specialization, aggregation, association, and cross-reference of dictionary elements and thus furnish a basis for the structured representation of complex engineering information.[20] Data stored or referenced by a dictionary is represented as a *content item* that is sub-typed according to different content representation types, for example a single value, range of values, table, etc. Content items have a relation to their description in the dictionary through the basic semantic unit identifier. The partition of content representation from the dictionary description provides a mechanism to define a reference model for components and to change the logical description (through product catalogue updates, etc.) without affecting the semantic definition. It also permits several publishers to cross-index their offerings with a reference model of their choice, for example the CSI MasterFormat classification system.

PLIB provides the resources to describe a component as a discrete object in a library and in terms of its placement in a product model or design context. The library structure defines a component in terms of three object topologies, a general model description (component feature and material definition), functional model description (context parameter definition) and associated views (functional model representation).

The dictionary elements that represent component attributes, called *properties*, have a relation to a *property data element type* (property_det) that defines a property's characteristics. Specialization of the property_det class to non dependent, context dependent, and dependent sub-types permits explicit differentiation of a component's discrete features and context parameters that are derived from its placement in a product model.

PLIB furnishes suitable representation structures for the IBES intermediate model and component repository because of the resources to define a semantic dictionary that describes components and functional views of components. We use the PLIB resources to describe and classify engineering analysis applications according to the tasks they automate as they relate to components. IBES also extends the PLIB library concept by adding the inference rules and intermediate model semantics to facilitate a general method for integration of an engineering application with a CAD system.

IBES Overview

IBES defines data types and provides rule-based inference to define an intermediate model that translates engineering content between a CAD model and a software application for component analysis. The intermediate model furnishes a common vocabulary to describe information from distinct knowledge sources, a product model, and a software application. This research assumes:

- A CAD environment that incorporates a standard product model representation providing a basis for agreement on product data semantics. Emerging product description standards are maturing to a level where their use is feasible in certain domains, for example the development of STEP models for the process industries.[19]

- A marketplace of Web-distributed software applications that automate and support engineering tasks across the facility life cycle.

Given these premises, the IBES model performs the following primary tasks:

A. Assists interpretation[5] of a product model to formulate data views for inputs to an engineering application. Assisted interpretation denotes computer-supported selection and extraction of component forms, functions and behaviors from a product model representation to a sharable view that furnishes the input for analysis and evaluation;
B. Validates the view data and software inputs to determine whether there is correct and sufficient information to run the service;
C. Provides a user interface that flags incomplete and mismatched data, and allows the user to take corrective action to modify property characteristics, and dynamically update the view validation status.
D. Furnishes a message interface to communicate with the remote application when it requests additional information for multi-step reasoning, and presents the request through a graphical user interface.

Figure 2 diagrams the IBES conceptual model framework. The clear boxes represent the primary objects of the intermediate model. They include the Application Requirement Template, Product Model View, Product Model Map, and Service Manager. The vectors in Figure 2 represent the tasks (**A**-**D**) described above. They include assisted interpretation (**A**) using a Map to derive a set of Product Model Views. Semantic validation (**B**) between an Application Requirement Template, (the specification of the inputs for a class of engineering application) and a set of Views. The Control step (**C** & **D**) represents the user-interaction processes to resolve incomplete information and supply additional data when requested by the application.

The conceptual framework to support tasks (**A)** and (**B)** was adapted from the heuristic classification method described by Clancey.[4] IBES also layers a human/computer interface over the inference framework to permit user interaction with the broker in support of tasks **(C)** and (**D).**

Clancy's study of expert system applications identifies a common modeling framework for problems that involve design and diagnosis. He notes that successful expert systems abstract domain data to a set of concepts that map to a hierarchy of pre-enumerated solutions by non-hierarchical, heuristic inference. The solutions are then refined within their hierarchy. Knowledge abstraction to a common, intermediate, and context independent model has also been explored in other research. For example Maluf and Wiederhold develop an ontology algebra that combines knowledge from distinct sources for interoperation and manipulation.[25]

Heuristic Classification	**IBES**
Domain data	Product model
Conceptual abstraction to problem class	*Assisted Interpretation to generate Product Model Views* that contain information selected from a product model and satisfy the information requirements of an *Application Requirement Template*
Heuristic match between problem class and pre-enumerated solution class	*Semantic validation* between the properties of an *Application Requirement Template* and the *properties* for a set of *Product Model Views*.
Refinement to specific solution	*Reference* from an *Application Requirement Template* to a specific engineering application that automates an engineering task.

Table 1: Correlation of heuristic classification terms with IBES objects

Table 1 relates heuristic classification terms to the IBES framework. Domain data correlates with a product model. The heuristic classification abstraction step correlates with assisted interpretation in which IBES selects and extracts product model information to a data view that satisfies the specification of an Application Requirement Template. Heuristic match correlates with the IBES semantic validation step in which the properties of a Product Model View are validated against their Application Requirement Template specification. The refinement step of heuristic classification correlates with the reference from an Application Requirement Template to a specific publisher's engineering application based on business and contractual requirements for a given project. IBES provides this reference, however it does not infer the application from business rules at this time.

Using IBES

This section describes how IBES works for the test case. The discussion follows the process diagram (Figure 3) that shows the flow of the IBES automation tasks (**A**-**D**) described previously.

It was noted above that the IBES framework depends on a standard product model description. Figure 4 shows the CAD representation of a fragment of the Process and Instrumentation Diagram (P&ID) used in the test case and the IBES Service Dialog box (described below). The product model underlying the CAD representation is based on a small subset of an object model, the Activity Resource Model (ARM), of the STEP standard for plant spatial configuration, Application Protocol 227 (AP227).[19] The IBES adaptation of the product model describes three object topologies, a functional description of a piping sub-system, the associated operating cases for the process material and design parameters, and the physical components (pipes, pumps, valves, etc.). It also provides relations between the physical and functional representations. The prototype includes essentially one piping sub-system composed of several pipe segments, a control valve, one stream definition and operating cases for minimum, maximum, and upset (abnormal) flow conditions. These topologies contain the information for an interpretation that selects and extracts information for control valve selection.

Assisted Interpretation

The IBES automation process begins with assisted interpretation (Figure 3, task **A**). The product model is the input for this task. We begin with a discussion of the task constraints, followed by the method, and end with a description of the task output, a set of Product Model Views.

The IBES system constrains assisted interpretation for a particular engineering task through a human computer interface for the CAD system that permits the user to pick a component and then an engineering application that is registered for the component class in the IBES library. For the test case the user selects a control valve symbol from the P&ID, and then the preliminary sizing application from a pop-up dialog box (Figure 4).

Once the system determines the class of engineering application selected by the user, it initializes the appropriate intermediate model instances to constrain and support an interpretation query. The query searches product model data structures for information that corresponds to the inputs of an engineering analysis application as specified by an Application Requirement Template. If the query finds a product model data item for an Application Requirement Template property specification, it translates the item to an appropriate Product Model View property content item and property det instance.

The upper portion of Figure 5 shows the IBES intermediate model class definitions that were introduced conceptually in Figure 2. The lower part of Figure 5 shows the object instances initialized for the test case. These objects are described in the following 3 sections.

- ❑ Application Requirement Template

An Application Requirement Template is a wrapper that defines the engineering function and specifies the data required by an application. In principle, the Application Requirement Template topology in the IBES library is structured according to a functional decomposition hierarchy of engineering tasks[27]. For the test case, the Application Requirement Template, *cvSelectionTemplate*, is simply a direct child of the IBES abstract class applicationRequirementsTemplate, which itself is a child of the PLIB dictionary_element class (not shown in Figure 5).

The value of the Application Requirement Template *definition* attribute describes the engineering function for an application. For the cvSelectionTemplate instance the value of this attribute is ‘Template for control valve sizing’. The list attribute *described_by* provides the specification of properties[22] required for control valve sizing.

Each *property* of an Application Requirement Template has a *code* (property name) and a relation to a property_det[22] instance that describes the *domain* and *unit* characteristics of the *property*. The *Inlet_Pressure* property instance in Figure 5 relates to a conditional property_det that has the value ‘real measure_type’ for *domain* and ‘psi’ for *unit*. The *Inlet_pressure* property instance relates to a conditional property_det because its value is a context parameter selected from the operating cases for the piping sub-system in the product model. If it had been a dependent_det, it would specify the property_det objects upon which it depends and the derivation function. Property characteristics furnish the semantic context for the validation of information selected from the product model, described in the Section titled “Validation.”

- Product Model Map

A Product Model Map provides two levels of constraint for assisted interpretation. First, it indicates to the system which query expressions to apply for a given product model schema and Application Requirement Template. Second, it is a control object for a query search.

The conceptual object diagram in Figure 2 indicates the coordination purpose of the Product Model Map class by showing it as a constraint object for the interpretation relation. Similarly the process diagram in Figure 3 shows the Product Model Map class as a constraint for assisted interpretation. Figure 6 indicates the constraint relationship explicitly for the test case. It shows how a Product Model Map connects the AP227 schema and the cvSelectionTemplate class with a query expression file. Thus a Product Model Map defines the applicable context of an interpretation query. A publisher would need to define a Map and a set of query expressions for each type of application multiplied by the number of standard product model schemata that it supports.

While query expressions depend on a product model’s data structure, constraints such as start and goal objects, and intermediate variables, can be abstracted and referenced as attributes of context independent control objects. The list attributes *query_variables* and *control_arguments* of the Product Model Map class serve this purpose. For example, the control valve instance selected by the user provided the initial state for the model search of the test case. Its identification attribute value was referenced in the *query_variables* attribute of the Product Model Map instance. To find inlet pressure data, the query searched the piping circuit upstream of the selected valve symbol, skipping intermediate objects such as a *pipe connector* until a *pipe segment* was identified for which *operating cases* that supplied inlet pressure information could be found. During the search, the Product Model Map referenced the *pipe connector* instance in the *control_arguments* attribute for comparison against the goal object of the query.

Product model query expressions provide the method to select and extract information for assisted interpretation (Figure 3, task **A**). Since the schema of a STEP model is statically defined, the query expressions for a particular interpretation may also be pre-determined.

❑ Product Model Views

The output of assisted interpretation is a set of Product Model Views that provide input data for the engineering analysis application (Figure 3, task **A**). Several Views are required by an application when a range of component behaviors must be computed for distinct design conditions. An Application Requirement Template specifies the default Views that are required by an application, however there are cases in which the product model contains incomplete or extra information. The broker handles these requirements and generates Views dynamically. We discuss how IBES handles incomplete information in the section titled “Assisted Interpretation.”

IBES initializes Product Model View instances by copying information from the CvSelectionTemplate instance. The description and number of default Product Model Views are obtained from a software interface object provided by an application publisher. Figure 5 shows a representative instance of a Product Model View with the value ‘v1’ for the *identified_by* attribute and ‘Minimum Flow’ for the *definition* attribute. The diagram also shows the *Inlet_Pressure* property instance and its relation to a content_item instance with the value ‘20.0’ for the *value* attribute. The diagram also shows the relation to a conditional_det instance that holds the property’s characteristics obtained from the product model.

Figure 7 shows the graphical user interface presented to the user after the system has queried the product model for the test case. The tree view control on the left displays the component family and specific class selected by the user. The tabbed dialog box displays five views generated by the system. Four of them are the default views required by the application for minimum, maximum, and upset material operating cases, and a view of normalized or common data. The fifth view, UPSET1, indicates that the query detected an additional operating case in the product model and IBES generated a view for it. The tabbed dialog box also shows the properties for the selected view, *Specific_Gravity*, *Stream_Phase*, etc., the characteristics of each property, and their validation status. Selecting a property from the list box lets a user inspect its characteristics in the edit controls. The characteristics for the *Inlet_Pressure* property are displayed.

Validation

The set of Product Model Views generated by a query provides the input for the property validation task. The Application Requirement Template specification provides the constraints, while the IBES validation algorithm furnishes the method (Figure 3, task **B**).

The IBES validation mechanism for assisted interpretation supports the viewpoint that most learned behavior, once mastered, becomes implicit for the performer. The user modifies a data item only when an interruption or breakdown[28] occurs from identifying incomplete or inaccurate information during interpretation, and when the application requires additional information to support multi-step reasoning. Otherwise, the interpretation activity itself is transparent for the user.

Before the user can invoke the application, each Product Model View *property* must satisfy its validation constraints. For example the *code*, *domain* and *unit* attributes for the *Inlet Pressure* property instance (Figure 5) must match the property specification defined by the Application Requirement Template instance.

Property characteristic validation assures that information obtained from the product model is relevant and sufficient for invoking the analysis application. To accommodate user-initiated changes to property characteristics, the validation process must also be dynamic.

The IBES implementation of the validation procedure is comprised of backward chaining rules that pattern match the data between the specification of an Application Requirement Template and its Product Model View *property characteristic* values. Since Application Requirement Template and Product Model View representations abstract information from an implementation context, the validation algorithm is general for any Product Model View-Application Requirement Template relation.

To control rule activation, IBES defines objects called, respectively, a property match controller, view match controller, and service match controller, that monitor the state of the validation process. A modification or change to the value of a characteristic changes an associated control object via dependency rules. The state of the control objects, in turn, governs the activation of the validation rules. The state-based nature of rule activation permits the validation process to be dynamic.

Figure 8 is a conceptual flow diagram of the IBES validation algorithm for the test case. Initial activation of the validation rule set begins with the goal to invoke the preliminary valve sizing application. A *service match controller* monitors the satisfaction of this goal by indicating whether all the Product Model View instances generated by the query 'match' the specification furnished by the cvSelectionTemplate instance. A specific Product Model View instance, for example 'v1' (Figure 5), is 'matched' when all its properties, for example the *Inlet_Pressure* property instance, 'match' their Application Requirement Template property specification. A *view match controller* monitors the satisfaction of this sub-goal for each Product Model View instance. Similarly, a property is validated when all its *characteristic* values match their specification. For example, the values 'real_measure_type' and 'psi' for the *domain* and *range* characteristics of the *Inlet_Pressure* property instance must match with their Application Requirement Template specification. A *property match controller* monitors the satisfaction of this sub-goal for each property instance.

The state-based nature of the algorithm allows for the easy identification of incomplete and mismatched data and the invocation of the application according to the successful match of the underlying data. The next section describes how the match rules are integrated with the user interface to enable user control of the broker.

User-initiated control

As discussed in previously, IBES supports user input to resolve incomplete information and provide additional information for multi-step reasoning (Figure 3, Tasks **C** & **D**). The following sections discuss how IBES meets these objectives.

- Incomplete Information

The IBES system must respond reasonably to incomplete information or information that does not match between a View and Application Requirement Template. An example of information mismatch would be metric versus English unit representation for the *Inlet_Pressure* property of the test case. The baseline response of IBES to incomplete information is to query the user and update the View.

As per the validation step, the IBES process for flagging incomplete information takes as input the set of Product Model Views formed by the query and is constrained by the data specification of the Application Requirement Template (Figure 3, Task **C**). The method for this process integrates the validation algorithm with a transaction model for changing property characteristics through the user interface. The output of this process is a validated and resolved set of Product Model View instances ready for transmission to the engineering analysis program.

For example, the possible validation states for the *Inlet_Pressure Unit* characteristic, 'psi', can be 'matched', 'not matched' and 'unknown'. The validation algorithm flags the *unit* characteristic 'matched' or 'not matched' depending whether it pattern matched with the Application Requirement Template specification for this characteristic. The Service Manager GUI (graphical user interface) (Figure 7) indicates the match status where the user can make modifications. The checked box for the *Inlet_Pressure Unit* characteristic indicates that the value 'psi' was matched. An unchecked box would indicate 'not matched', while a grayed check box would indicate the match status is 'unknown'. After making a modification, for example by changing the *Unit* value from 'psi' to 'N/m^2', the match status changes to 'unknown'. When the user presses the Apply button the system commits the latest changes to the underlying characteristic instances. As discussed above, doing this causes the validation rules to activate and re-validate the Product Model View data.

- Multi-step reasoning

Often analysis applications perform multi-step reasoning and require additional data from the subscriber (Figure 3, Task D). This requirement indicates a need for a communication language that supports dynamic exchange of structured information like STEP data, and associated format information to enable appropriate presentation to the user.

IBES addresses this requirement for the valve selection test case with a message interface that permits limited two-way communication. After opening a connection to a server, IBES can send two messages, a Service Request and an Additional Data Response. It also receives two messages, a New Data Request and Report Transmission. When IBES sends a Service Request it transmits View data to the server for processing. It sends an Additional Data Response after it receives an Additional Data Request from the server and queries the user for the information. Currently, this function is available only for querying a single attribute value per message. It is used in the valve selection test case when the remote service queries the user to obtain a value for the valve port nominal diameter. When IBES receives a Report Transmission, it presents the information to the user through the Service Manager GUI.

Implementation

The IBES prototype currently consists of three sub-systems, a CAD/Product model, IBES model, and analysis applications residing on a server. The product model and IBES model knowledge bases are written using Rete++[11], an inference engine and Windows 95™ C++ application programming interface that extends the 'C' Language Integrated Production System (CLIPS).

Figure 10 shows the logical model architecture. To expedite implementation, the symbolic model elements (product model and IBES module) use blackboard architecture instead of inter-process communication. Nonetheless the system defines and invokes each module (objects, methods, and rule sets) separately.

The CAD subsystem consists of AutoCAD®, with an AutoCAD ARX customization. The ARX program controls the programmatic interface to the CAD system and communicates with the other modules.

The analysis application for the control valve selection problem is a small program that takes as input the validated View data sent by the broker and returns a list of control valves by computing the minimum and maximum coefficient of flow and looking up candidate valves in a database. The application is implemented in Java and resides on a server, also implemented in Java.

Discussion

This paper defines a conceptual framework to select, extract, validate, and communicate component information from a product model for an engineering analysis application. We test the hypothesis that these services can be abstracted for a general approach that supports the interoperation of Web-distributed analysis applications, and thus improves knowledge sharing between organizations. Preliminary evidence, based upon a proof-of-concept implementation and a test case, suggests that a general approach is possible.

The control valve problem selected for the test case addresses component configuration for the detailed design phase of a facility. Clearly more work is required to extend and test the generality of the framework for a range of information modeling conditions that affect view generation and match against application input specifications. The following categories highlight potential areas for further research:

Model query

It was noted that the interpretation query expressions for the test case were pre-defined. It was possible to pre-define the query because the data structures for the product model were static. This simplifying assumption would need to be relaxed for a model that supports schema evolution for which it would be necessary to construct the interpretation query dynamically. Furthermore, the IBES architecture currently supports the formulation of data views that abstract and simplify the product model representation for an engineering analysis application. Other analysis applications require model elaboration and increased representational complexity, or the translation of one model view to another more suited to a particular analysis requirement. For example, applications for cost estimating and construction planning may require additional data to perform reasoning. Future research should explore formal approaches to model formulation[24] and query planning for model query and interpretation.

Incomplete information

The incomplete information problem poses significant database design issues. A fully implemented system must provide a mechanism to add, populate, modify, and delete schemata in the product model that are required for Product Model View formulation. In addition it would need to manage the preservation of new data for incomplete or partial transactions involving Product Model Views. While IBES maintains new information in the Product Model View, it does not currently address dynamic changes to the underlying product model. Other research, for example EDM-2[9], addresses such issues.

Communication language

A standard language for communication between software objects is the subject of considerable research and development. The Agent Communication Language (ACL) based on the Knowledge Interchange Format (KIF) vocabulary fulfills this function.[12] Similarly the emergent CORBA standard used in conjunction with the proposed eXtended Markup Language (XML) or an equivalent could provide these services.[15]

Conclusion

Our focus on modeling engineering content in support of specific analysis tasks emphasizes practice-centric rather than theoretic engineering research methodology. The strength of this approach is that it positions this research contribution closer to the problems encountered by the AEC industry.

We feel the practice-centric approach can contribute to the standardization efforts for describing product data. By formally modeling the input specifications for engineering applications, IBES establishes a mechanism to test whether a product model schema supports the reasoning we wish to perform with the model. This type of feedback mechanism is critical for the development of a modeling testbed.

We have attempted to bridge the methodological gap with formal theoretic approaches by founding the conceptual model for IBES on established knowledge engineering theory. However we recognize and have noted that the research would benefit from contributions from other domains.

To bridge our work with industry concerns and information technology strategies, we have positioned the research to address the critical business problem of sharing knowledge between information publishers and subscribers. We have also conceptualized the IBES integration services as middle-ware that could be implemented in an interoperation framework such as the CORBA standard and/or Microsoft DCOM/ActiveX.

Acknowledgements

This project is co-sponsored by a seed project award from the Center for Integrated Facilities Engineering (CIFE), Stanford University, and by Bechtel Engineering Company. We would like to extend our thanks to Carlos Elizondo of Bechtel for his generous support and assistance. Additionally, the author wishes to thank Professor Martin Fischer for his valuable comments and guidance.

References

1. Amor, R., Hosking, J., Groves, L., and Donn, M., Design Tool Integration: Model Flexibility for the Building Profession, *Proceedings of Building Systems Automation – Integration*. Madison, WI: University of Wisconsin-Madison, 1993.
2. Arnold, J. A., A Study of the Information Requirements for an Information Model of the Components that are Incorporated in Process Facilities, Unpublished Engineer Thesis, Department of Civil Engineering, Stanford University, 1996.
3. Bjrk, B.C., Basic Structure of a Proposed Building Product Model, *Computer-Aided Design*, 21(2): 71-78. London: Butterworth Scientific, Ltd., 1989.
4. Clancey, W. J., Heuristic Classification, *Artificial Intelligence* 27, Elsevier Publishers B.V. (North-Holland), pp. 289-350, 1985.
5. Clayton, M.J., Fruchter, R., Krawinkler, H., and Teicholz, P., Interpretation Objects for Multi-Disciplinary Design, *Artificial Intelligence in Design*, '94, ed. J.S.Gero and F. Sudweeks. Dordrecht, the Netherlands: Kluwer Academic Publishers, 1994.
6. Danner, B., Developing APs using the Architecture and Methods of STEP: Fundamentals of the STEP Methodology. Computer Integrated Construction Group, Building and Fire Research Laboratory, National Institute of Standards and Technology, U.S. Department of Commerce, ISO TC184/SC4/WG10 N74, 1996.
7. Driskell, L., Control-Valve Selection and Sizing, © Instrument Society of America 1983. Publishers Creative Services, Inc., ISBN 0-87664-628-3.
8. Eastman, C. M., Chase, S.C., and Assal, H.H., System Architecture for Computer Integration of Design and Construction Knowledge, *Automation in Construction* 2(2): 95-107. Amsterdam: Elsevier Science Publishers, 1993.
9. Eastman, C. M., Managing Integrity in Design Information Flows, *Computer Aided Design*, IPC Press, 1995.
10. El-Bibany, H., Architecture for Human-Computer Design, Management and Coordination in a Collaborative AEC Environment, Technical Report Number 70. Stanford, CA: Center for Integrated Facilities Engineering, 1992.
11. Forgy, C., Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, *Artificial Intelligence*, 19, pp. 17-37, 1982.
12. Genesereth, M.R., Singh, N.P., Syed, M.A., A Distributed and Anonymous Knowledge Sharing Approach to Software Implementation, Stanford University Department of Computer Science, Stanford, CA, 1994.
13. http://logic.stanford.edu/kif/specification.html

14. http://phrames.informatik.tu-muenchen.de/javasdai/v1/index.html

15. http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm

16. http://www.omg.org

17. http://www.microsoft.com

18. ISA-S75.01, Standards and Recommended Practices for Instrumentation and Control, 11th edition ©1991. Instrument Society of America, 67 Alexander Dr., P.O. Box 12277, Research Triangle Park, North Carolina 27709.

19. ISO CD 10303-227, International Standards Organization, Technical Committee ISO 184, Industrial automation systems and integration, Subcommittee SC 4, Industrial data and global manufacturing programming languages - ISO 10303 Product data representation and exchange - Part 227: Plant Spatial Configuration, 1994.

20. ISO CD 13584-10, International Standards Organization, Technical Committee ISO 184, Industrial automation systems and integration, Subcommittee SC 4, Industrial data and global manufacturing programming languages - ISO 13584, Parts Library - Part 10: Conceptual Descriptions: Conceptual Model of Parts Library, 1995.

21. ISO CD 13584-24, International Standards Organization, Technical Committee ISO 184, Industrial automation systems and integration, Subcommittee SC 4, Industrial data and global manufacturing programming languages - ISO 13584, Parts Library - Part 24: Logical Model of Supplier Library, 1995.

22. ISO CD 13584-42, International Standards Organization, Technical Committee ISO 184, Industrial automation systems and integration, Subcommittee SC 4, Industrial data and global manufacturing programming languages - ISO 13584, Parts Library - Part 42: Methodology for Structuring Part Families, 1995.

23. ISO WD TC/184/SC4/WG5 Working Draft, EXPRESS-X Reference Manual. Lab for Industrial Information Infrastructure, Rensselaer Polytechnic Institute, CII Building, Room 7015, Troy, New York 12180-3590 USA, 5/28/1996.

24. Levy, Y.L., Iwasaki, Y., Fikes, R., Automated Model Selection for Simulation Based on Relevance Reasoning, Knowledge Systems Laboratory, Stanford University, 701 Welch Road, Bldg. C, Palo Alto, CA 94304, 1996.

25. Maluf, D., Wiederhold, G., Abstraction of Representation for Interoperation, Department of Computer Science, Stanford University, Stanford, CA 94305

26. Wiederhold, G., Interoperation, Mediation, and Ontologies, Department of Computer Science, Stanford University, Stanford, CA94305.

27. Willems, P., A Functional Network for Product Modeling, PLI-88-16, IBBC-TNO, Rijkswijk, Netherlands, 1988.

28. Winograd, T., Flores, F., Understanding Computers and Cogniti*on, A New Foundation for Design*. Addison-Wesley Publishing Company © 1986, 1987 Ablex Publishing Corporation. ISBN: 0-201-11297-3, 1987.

Figure Captions:

Figure 1: Conceptual IDEF0 Diagram of IBES Integration Activity.

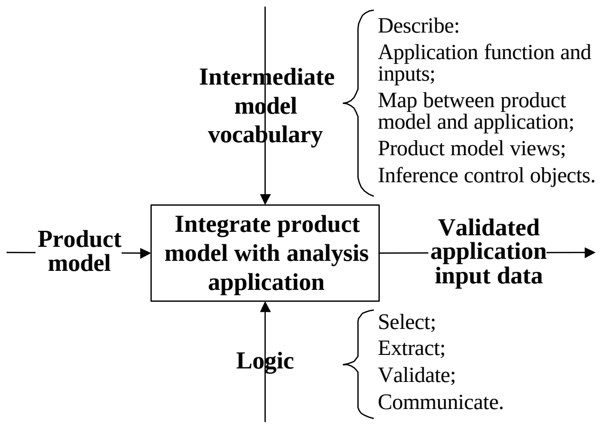

Intermediate model vocabulary
Describe:
Application function and inputs;
Map between product model and application;
Product model views;
Inference control objects.
Product model
Integrate product model with analysis application
Validated application input data
Logic
Select;
Extract;
Validate;
Communicate.

Figure 2: IBES intermediate model conceptual framework for information abstraction.

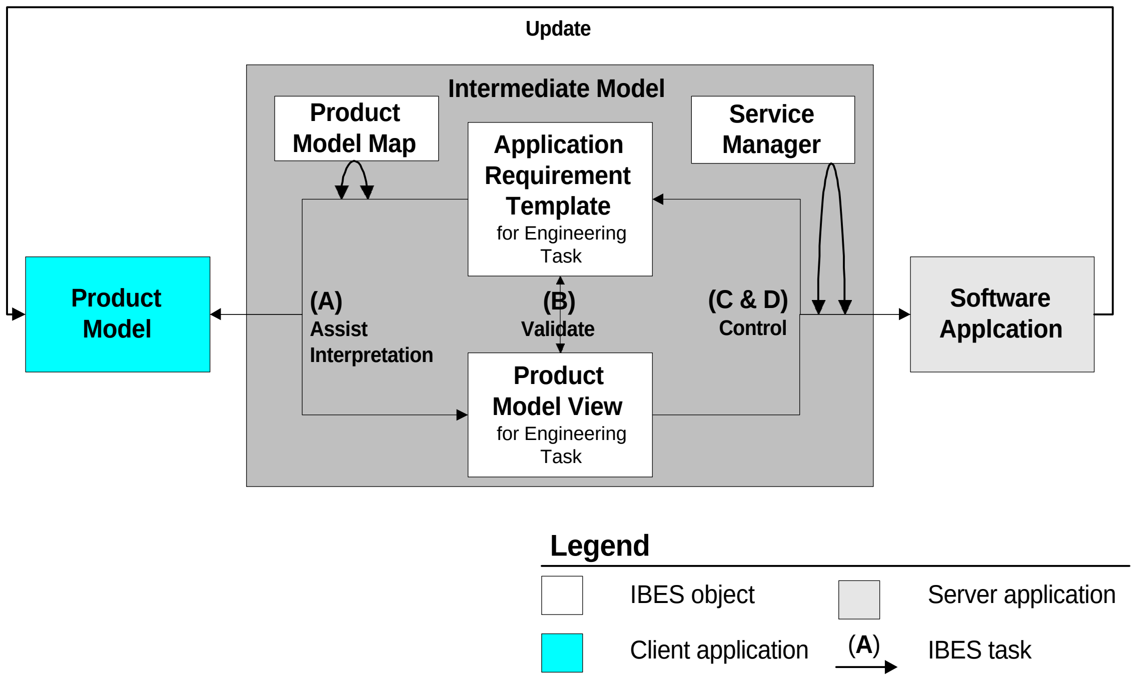
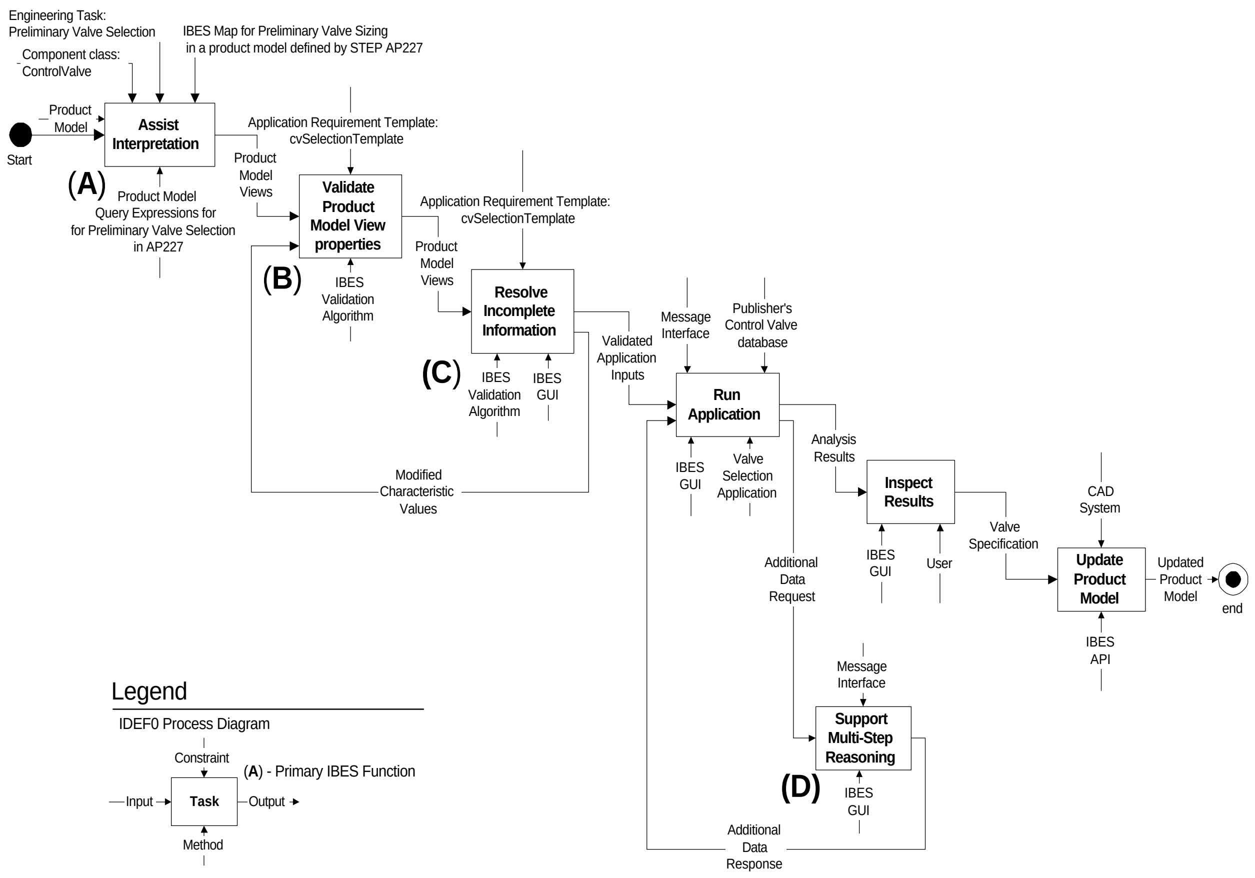
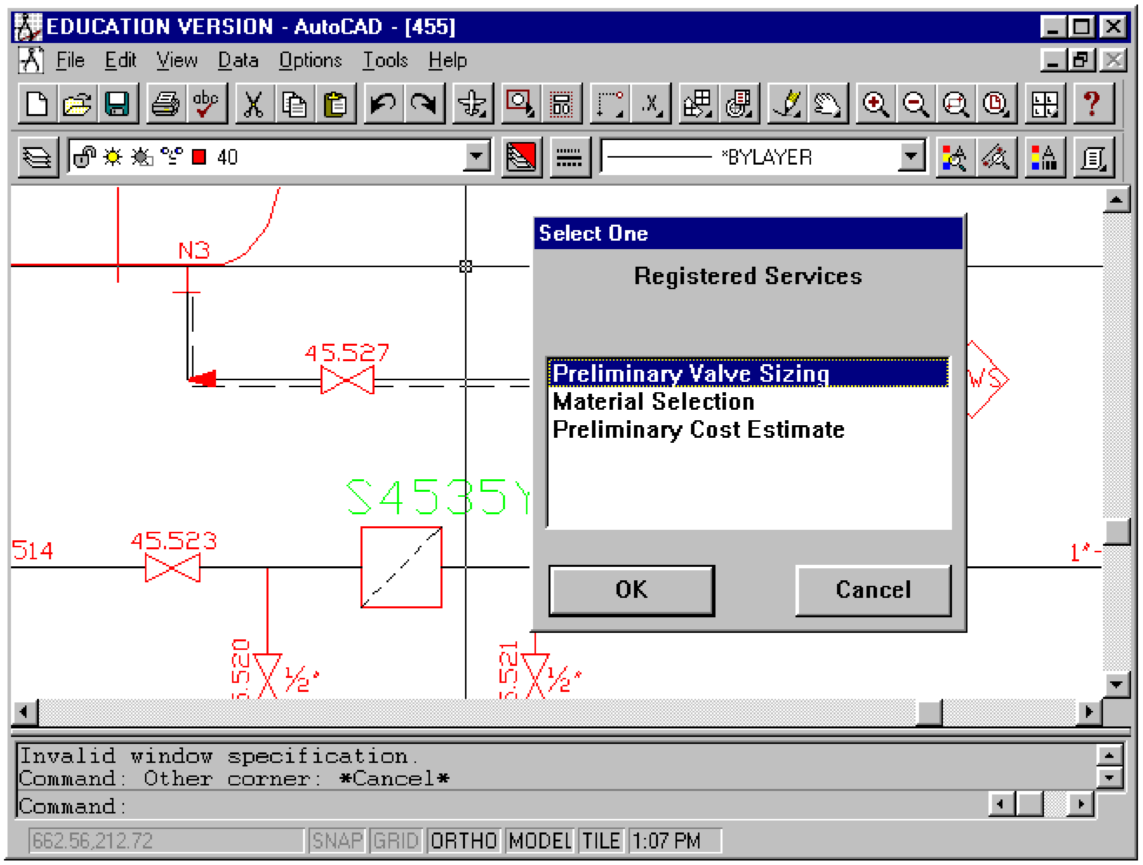


Figure 3: IDEF0 diagram of process automation tasks supported by IBES.



Engineering Task: Preliminary Valve Selection
IBES Map for Preliminary Valve Sizing in a product model defined by STEP AP227
Component class: ControlValve
Product Model
Start
Assist Interpretation
(A)
Product Model Query Expressions for for Preliminary Valve Selection in AP227
Application Requirement Template: cvSelectionTemplate
Product Model Views
Validate Product Model View properties
(B)
IBES Validation Algorithm
Application Requirement Template: cvSelectionTemplate
Product Model Views
Resolve Incomplete Information
(C)
IBES Validation Algorithm
IBES GUI
Modified Characteristic Values
Validated Application Inputs
Message Interface
Publisher's Control Valve database
Run Application
IBES GUI
Valve Selection Application
Analysis Results
Inspect Results
IBES GUI
User
Additional Data Request
Valve Specification
CAD System
Update Product Model
IBES API
Updated Product Model
end
Message Interface
Support Multi-Step Reasoning
(D)
IBES GUI
Additional Data Response
Legend
IDEF0 Process Diagram
Constraint
(A) - Primary IBES Function
Input
Task
Output
Method

Figure 4: Test case process plant sub-system, represented in AutoCAD as a process and instrumentation diagram (P&ID).


EDUCATION VERSION - AutoCAD - [455]
File Edit View Data Options Tools Help
40
BYLAYER
N3
45.527
S4535Y
514
45.523
WS
1"
520
½"
521
½"
662.56,212.72
SNAP GRID ORTHO MODEL TILE 1:07 PM

Select One

Registered Services

Preliminary Valve Sizing
Material Selection
Preliminary Cost Estimate

OK Cancel

```
Invalid window specification.
Command: Other corner: *Cancel*
Command:
```

Figure 5: IBES intermediate model objects

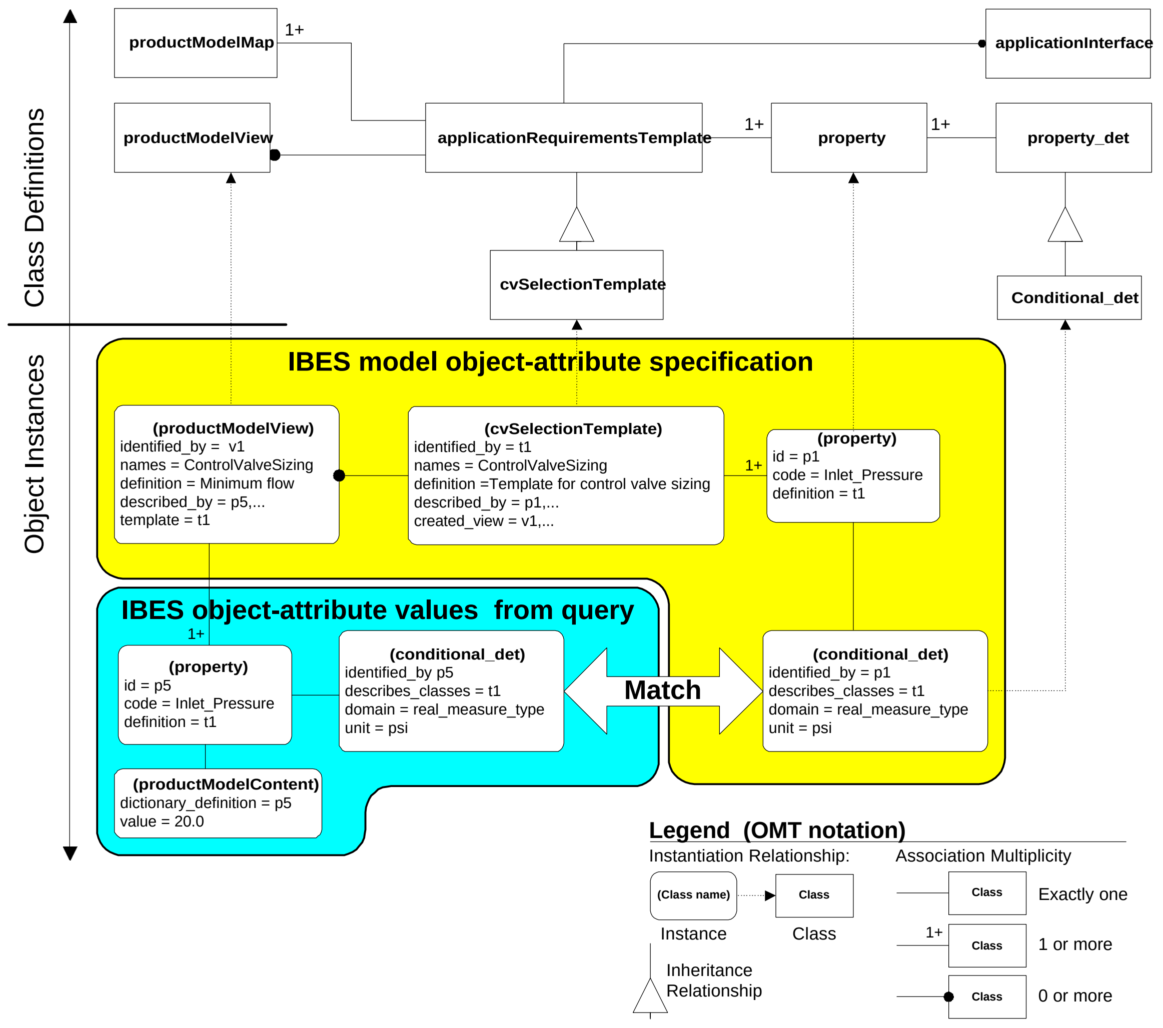


Figure 6: IBES Map relations for the test case.

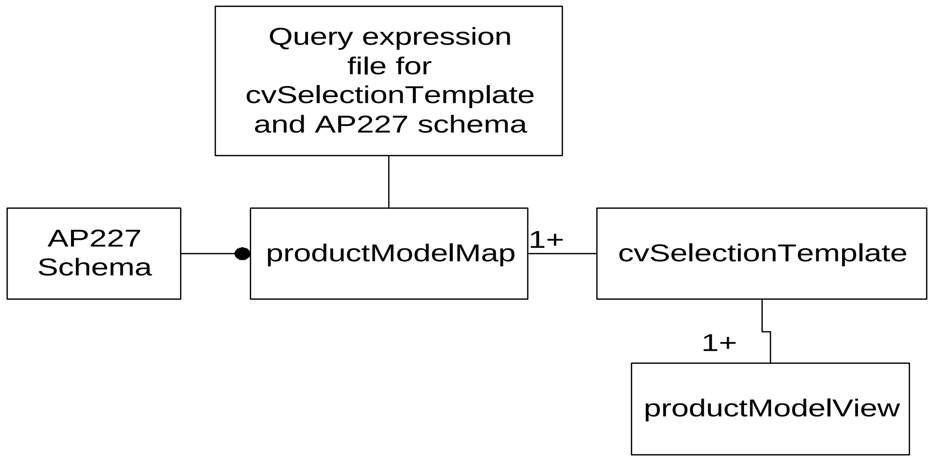
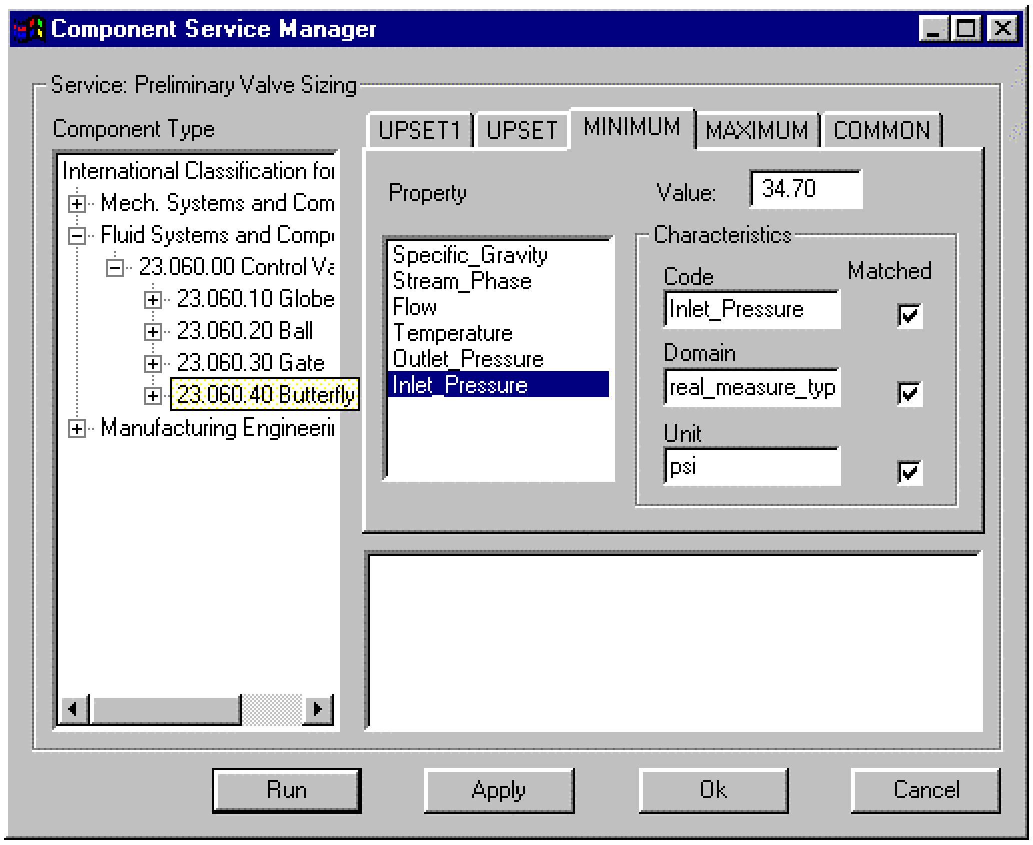


Figure 7: IBES component service manager graphical user interface.



Service: Preliminary Valve Sizing

Component Type

International Classification fo
Mech. Systems and Com
Fluid Systems and Comp
23.060.00 Control Va
23.060.10 Globe
23.060.20 Ball
23.060.30 Gate
23.060.40 Butterfly
Manufacturing Engineeri

Property: Specific_Gravity, Stream_Phase, Flow, Temperature, Outlet_Pressure, Inlet_Pressure

Value: 34.70

Characteristics

	Matched
Code: Inlet_Pressure	✓
Domain: real_measure_typ	✓
Unit: psi	✓

Figure 8: Conceptual flow diagram of data validation process.

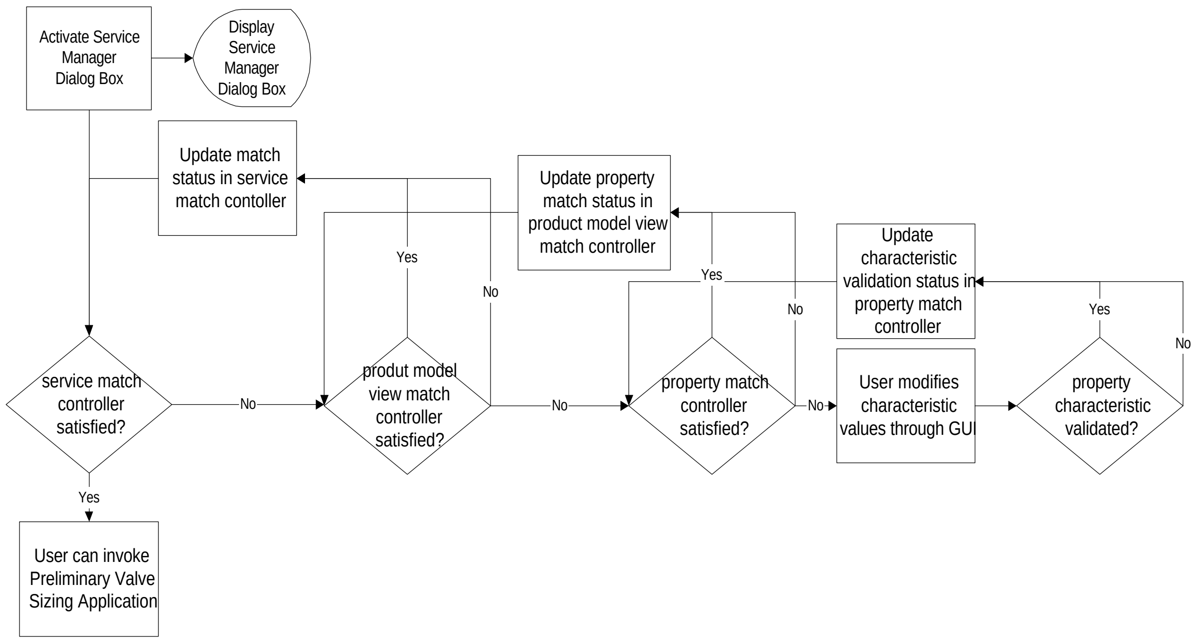
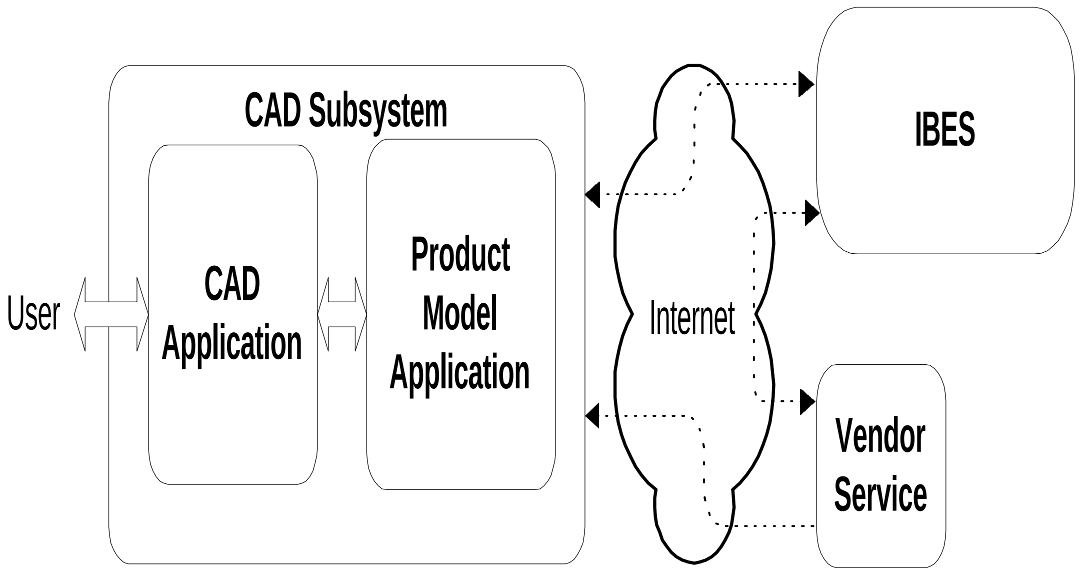

Activate Service Manager Dialog Box
Display Service Manager Dialog Box
Update match status in service match contoller
Update property match status in product model view match controller
Update characteristic validation status in property match controller
service match controller satisfied?
produt model view match controller satisfied?
property match controller satisfied?
User modifies characteristic values through GUI
property characteristic validated?
User can invoke Preliminary Valve Sizing Application
Yes
No

Figure 9: IBES prototype logical model.


CAD Subsystem
User
CAD Application
Product Model Application
Internet
IBES
Vendor Service